Refine Search

Search Results -

Term	Documents
FILTER	1561277
FILTERS	505581
ITEM	388001
ITEMS	461585
(2 AND (ITEM NEAR FILTER)).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	2
(L2 AND (FILTER NEAR ITEM)).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	2

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L14

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, September 22, 2006 **Purge Queries** Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L14	l2 and (filter near item)	2	L14
L13	(6119129.pn.) and filter	0	L13
L12	(6016497.pn.) and filter	0	L12
L11	(5970497.pn.) and filter	0	L11
L10	(5970497.pn.) and filter	0	L10
L9	(5878415.pn.) and filter	1	L9
L8	(5412804.pn.) and filter	0	L8

L7	(6112198.pn.) and filter	1	L7
L6	L4 and filter	2	L6
L5	L4 and expand$	0	L5
L4	6356892.pn.	2	L4
L3	L2 and (service near query)	1	L3
L2	expand$ near filter	2041	L2
L1	query near expand$	380	L1

END OF SEARCH HISTORY

L2: Entry 1 of 1 File: USPT Mar 12, 2002

DOCUMENT-IDENTIFIER: US 6356892 B1
TITLE: Efficient implementation of lightweight directory access protocol (LDAP) search queries with structured query language (SQL)

Abstract Text (1):
A method of hierarchical LDAP searching in an LDAP directory service having a relational database management system (DBMS) as a backing store. The method begins by parsing an LDAP filter-based query for elements and logical operators of the filter query. For each filter element, the method generates an SQL subquery according to a set of translation rules. For each SQL subquery, the method then generates a set of entry identifiers for the LDAP filter query. Then, the SQL subqueries are combined into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the LDAP filter query.

Brief Summary Text (9):
LDAP provides a number of known functions including query (search and compare), update, authentication and others. The search and compare operations are used to retrieve information from the database. For the search function, the criteria of the search is specified in a search filter. The search filter typically is a Boolean expression that consists of qualifiers including attribute name, attribute value and Boolean operators like AND, OR and NOT. Users can use the filter to perform complex search operations. One filter syntax is defined in RFC 2254.

Brief Summary Text (12):
The present invention addresses the problem of efficiently mapping an LDAP filter into an SQL query.

Brief Summary Text (14):
It is a primary object of this invention to provide a method for searching a relational database using hierarchical, filter-based queries, such as LDAP.

Brief Summary Text (15):
Another primary object is to provide an algorithm that combines basic LDAP filter expressions into a preferably single SQL query that retrieves target entries that exactly match given search criteria.

Brief Summary Text (20):
It is another more specific object of the present invention to provide a recursive algorithm that can deal with LDAP filter operators in a consistent way, and that deals with complicated LDAP queries with infinite number of logical operators.

Brief Summary Text (23):
The present invention solves the problem of efficiently mapping an LDAP filter into an SQL query using unique entry identifier (EID) sets. According to the inventive method, a SQL subquery is first generated for each LDAP operator based on given translation rules. The SQL subquery generates a set of entry EIDs that match the LDAP basic operation. Thereafter, the SQL subqueries are combined into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the LDAP filter query. Thus, for example, if the LDAP logical operator is OR(.vertline.), the invention then preferably uses an SQL UNION to union the sets generated from the subquery. If the LDAP logical operator is AND (&), the invention

preferably uses an SQL INTERCEPT to intercept the sets generated from the subquery. If the LDAP logical operator is NOT, the invention preferably excludes entries by negating the IN operation before the subquery. Thus, the combination rules includes, for example, mapping the LDAP logical OR operation to an SQL UNION, mapping the LDAP logical operation AND to SQL INTERCEPT, and mapping the LDAP logical operation NOT to SQL NOT IN.

Brief Summary Text (24):
Generalizing, according to the preferred embodiment, a method for searching a relational database using hierarchical, filter-based queries begins by parsing a filter-based query for elements and logical operators of the filter query. For each filter element, the method generates an SQL subquery according to a set of translation rules. For each SQL subquery, the method then generates a set of entry identifiers for the filter query. Then, the SQL subqueries are combined into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the filter query.

Drawing Description Text (7):
FIG. 5 is a simplified flowchart of the inventive method for hierarchical LDAP searching in an LDAP directory service having a relational database management system as a backing store using LDAP filter queries mapped efficiently to SQL; and

Detailed Description Text (10):
One of ordinary skill should appreciate that the system architectures illustrated in FIGS. 4A-4C are not to be taken as limiting the present invention. The inventive technique may be used to search any relational database using hierarchical, filter-based database queries.

Detailed Description Text (11):
The technique is now described generally with reference to the flowchart of FIG. 5. This diagram illustrates the basic method of the invention to provide hierarchical LDAP searching in an LDAP directory service having a relational database management system (DBMS) as a backing store. The method begins at step 60 by parsing an LDAP filter-based query for elements and logical operators of the filter query. For each filter element, the method continues at step 62 to generate an SQL subquery according to a set of translation rules, which will be defined below. For each SQL subquery, the method then continues at step 64 to generate a set of unique entry identifiers for the LDAP filter query. Then, at step 66, the SQL subqueries are combined into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the LDAP filter query. At step 68, the single SQL query is applied to the database step. Results are returned at step 70.

Detailed Description Text (57):
In the above representation, <table list> and <where expression> are the two null terminated strings returned by the SQL generator. The <root dn id> is the unique identifier of the root dn. The where clause should only be generated if <where expression> is not the empty string and no errors where detected in the parsing the LDAP filter.

Detailed Description Text (58):
As is well-known, LDAP search queries comprise six basic filters with the format <attribute> <operator> <value>. Complex search filters are generated by combining basic filters with Boolean operators AND (&), OR (.vertline.) and NOT (!).

Detailed Description Text (60):
As described above in the flowchart of FIG. 5, an SQL subquery is generated for each LDAP filter element according to a set of translation rules. This was step 62 in the method. The following sets forth preferred translation rules used to generate SQL queries for basic LDAP filters according to the present invention:

Detailed Description Text (62):
LDAP filter

Detailed Description Text (67):
LDAP filter

Detailed Description Text (74):
LDAP filter

Detailed Description Text (80):
LDAP filter

Detailed Description Text (84):
As described above, according to the inventive method, for each LDAP filter element or sub-expression, there is a set of entries (EIDs) that will satisfy the element. Thus, each element generally maps to a set of EIDs. The EID sets are then merged together, preferably into a single SQL query, using a set of combination rules. Thus, if a pair of LDAP filter elements are subject to an LDAP logical OR operator, the corresponding EID sets are merged using an SQL UNION logical operator. If a pair of LDAP filter elements are subject to an LDAP logical AND operator, the corresponding EID sets are merged using an SQL INTERSECT logical operator. If a pair of LDAP filter elements are subject to an LDAP logical NOT operator, the corresponding EID sets are merged using an SQL NOT IN logical operator. As will also be seen, these combination rules are applied recursively such that all LDAP elements associated with a particular logical operator are processed into the SQL query. This recursive processing facilitates handling of even complicated LDAP queries having numerous layers of logical depth.

Detailed Description Text (86):
With reference to FIG. 6A, the method begins at step 72 by concatenating "(" to the generated SQL query (which, in the first iteration, is otherwise a skeleton query). The routine then tests at step 74 to determine whether the LDAP filter element includes the AND logical operator. If the outcome of the test at step 74 is positive, the AND logical operator is present. Typically, an AND logical operator sets apart at least a pair of subexpressions, and thus the routine includes appropriate logic to handle each subexpression separately in a recursive manner. To this end, the routine continues at step 76 and sets an process variable nextFilter equal to ProcessFilter(nextFilter). At step 78, the routine concatenates the SQL INTERSECT operator to the SQL query. This step maps the LDAP AND logical operator in the manner previously described. At step 80, a test is made to determine whether nextFilter is nil, i.e. whether all subexpressions associated with the AND logical operator in the LDAP filter have been processed. If not, the routine returns at step 82. If, however, the outcome of the test at step 80 is negative, the routine loops back to step 76 to process the next subexpression associated with the AND logical operator currently being mapped.

Detailed Description Text (87):
If the outcome of the test at step 74 is negative, which indicates that the AND logical element is not in the filter element, the routine branches to step 84 to determine whether the LDAP OR logical operator is present. If the outcome of the test at step 84 is negative, the routine branches to FIG. 6B. If, however, the outcome of the test at step 84 is positive, the routine continues at step 86 with a recursive call into the ProcessFilter(nextFilter) subroutine previously described. Just as with the AND logical operator, an OR logical operator will typically include subexpressions that must be processed in a recursive manner.

Detailed Description Text (89):
If the LDAP filter element includes neither AND nor OR, the routine continues at step 92 to determine whether the NOT logical operator is present. If so, the routine continues at step 94 to add the NOT IN logical operator to the SQL expression being generated. The routine then continues at step 96 to enter the recursive call so that all associated subexpressions may be parsed through the algorithm in the manner previously described. Thus, at step 98, a test is performed to determine whether all subexpressions associated with the NOT operator have been processed. If so, the routine returns at step 100; otherwise, the routine loops back to step 96 and processes the next subexpression.

Detailed Description Text (90):
The remainder of the flowchart describes the mapping of simple filter elements using the set of translation rules previously described. At step 102, a test is made to determine whether the LDAP filter includes a simple equality statement (e.g., "a=1"). If so, the routine continues at

step 104 to concatenate the associated attribute column name value into the SQL query. Thereafter, or if the result of the test at step 102 is negative, the routine continues in FIG. 6C.

Detailed Description Text (91):
At step 106, a test is made to determine whether the LDAP filter element is a substring filter. If so, the routine continues at step 108 to concatenate the associated attribute column name value into the SQL query. Thereafter, or if the result of the test at step 106 is negative, the routine continues at step 110 to determine whether the LDAP filter element is a greater than or equal expression. If so, the routine continues at step 112 to concatenate the associated attribute column name substring value into the SQL query being constructed. Thereafter, the routine continues at step 114 to determine whether the LDAP filter element is a less than or equal to expression. If so, the routine concatenates the associated value into the SQL expression at step 115 and continues at step 116. Step 116 is reach also as a result of a negative outcome of step 114.

Detailed Description Text (92):
At step 116, a test is performed to determine whether the LDAP filter expression is a simple exists filter. If so, the routine concatenates the associated value into the SQL expression at step 118. The routine then continues with step 120 in FIG. 6D, which is also reached by a negative outcome to the test at step 116. Step 120 tests whether the LDAP filter includes the simple approximate filter. If so, the appropriate value is concatenated into the SQL query at step 122. At step 124, the SQL query is closed by concatenating an ")" value to complete the processing.

Detailed Description Text (94):
BEGIN PROCESSFILTER ALGORITHM ON AN LDAP-FILTER.

Detailed Description Text (97):
LDAP FILTER:

Detailed Description Text (98):
1) For complex ldap_filter with an AND operation:

Detailed Description Text (104):
2) For complex ldap-filter with an OR operation:

Detailed Description Text (110):
3) For complex ldap-filter with a NOT operation:

Detailed Description Text (113):
4) For simple equality filter:

Detailed Description Text (117):
5) For simple substring filter:

Detailed Description Text (121):
6) For simple greater or equal filter:

Detailed Description Text (125):
7) For simple less or equal filter:

Detailed Description Text (129):
8) For simple attribute exists filter:

Detailed Description Text (132):
9) For simple approximate filter:

Detailed Description Text (137):
Return next - ldap-filter.

Detailed Description Text (140):
With the basic translation rules and the EID sets approach described in the flowcharts of FIG. 5 and FIGS. 6A-6D, the following are the SQL queries that the invention generates for a representative LDAP filter query of the following form (.vertline.(f1=`v1`) (f2=`v2`)):

Detailed Description Text (172):
Filter String:

Detailed Description Text (186):
Filter String:

CLAIMS:

1. A method for searching a relational database using hierarchical, filter-based queries, comprising the steps of:

parsing a filter-based query for elements and logical operators of the filter query;

for each filter element, generating an SQL subquery according to a set of translation rules;

for each SQL subquery, generating a set of entry ID's for the filter query; and

combining the SQL subqueries into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the filter query.

2. The method as described in claim 1 wherein the filter-based query is a Lightweight Directory Access Protocol (LDAP) directory service query.

3. The method as described in claim 2 wherein the logical operators of the LDAP filter-based query include AND, OR and NOT.

10. The method as described in claim 1 wherein the step of combining the SQL subqueries is carried out recursively until all filter elements of the filter query have been processed into the SQL query.

11. A method for searching a relational database from a Lightweight Directory Access Protocol (LDAP) directory service generating filter-based queries, comprising the steps of:

parsing an LDAP filter-based query for elements and logical operators of the LDAP filter query;

for each LDAP filter element, generating an SQL subquery according to a set of translation rules;

for each SQL subquery, generating a set of entry ID's for the LDAP filter query; and

combining the SQL subqueries into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the LDAP filter query.

12. The method as described in claim 11 wherein the logical operators of the LDAP filter-based query include AND, OR and NOT.

16. The method as described in claim 11 wherein the step of combining the SQL subqueries is carried out recursively until all filter elements of the filter query have been processed into the SQL query.

17. A computer program product in computer-readable media for searching a relational database using hierarchical, filter-based queries, comprising:

means for parsing a filter-based query for elements and logical operators of the filter query;

means for generating an SQL subquery for each filter element according to a set of translation rules;

means for generating a set of entry ID's for each SQL subquery; and

means for combining the SQL subqueries into a single SQL query according to a set of combination rules chosen corresponding to the logical operators of the filter query.

18. The computer program product as described in claim 17 wherein the filter-based query is a Lightweight Directory Access Protocol (LDAP) directory service query.

19. The computer program product as described in claim 18 wherein the logical operators of the LDAP filter-based query include AND, OR and NOT.

24. A directory service, comprising:

a directory organized as a naming hierarchy having a plurality of entries each represented by a unique identifier;

a relational database management system having a backing store for storing directory data;

means for searching the directory, comprising:

means for parsing an hierarchical, filter-based query for elements and logical operators of the filter query;

means for generating a relational database subquery for each filter element according to a set of translation rules;

means for generating a set of unique identifiers for each relational database subquery; and

means for combining the relational database subqueries into a single relational database query according to a set of combination rules chosen corresponding to the logical operators of the filter query.

27. In a directory service having a directory organized as a naming hierarchy, the hierarchy including a plurality of entries each represented by a unique identifier, the improvement comprising:

a relational database management system having a backing store for storing directory data;

means for searching the directory, comprising:

means for parsing an hierarchical, filter-based query for elements and logical operators of the filter query;

means for generating a relational database subquery for each filter element according to a set of translation rules;

means for generating a set of unique identifiers for each relational database subquery; and

means for combining the relational database subqueries into a single relational database query according to a set of combination rules chosen corresponding to the logical operators of the filter query.

Previous Doc Next Doc Go to Doc#